Evolutionary Cost-cognizant Test Case Selection and Prioritization for Object-Oriented Programs

Abdulkarim Bello, Abu Bakar Md. Sultan, Abdul Azim Abdul Ghani, Hazura Zulzalil

Abstract*: Test case prioritization (TCP) is a software testing technique that finds an ideal ordering of test cases for regression testing, so that testers can obtain the maximum benefit of their test suite, even if the testing process is stop at some arbitrary point. The recent trend of software development uses OO paradigm. This paper proposed a cost-cognizant TCP approach for object-oriented software that uses path-based integration testing. Path-based integration testing will identify the possible execution path and extract these paths from the Java System Dependence Graph (JSDG) model of the source code using forward slicing technique. Afterward evolutionary algorithm (EA) was employed to prioritize test cases based on the severity detection per unit cost for each of the dependent faults. The proposed technique was known as Evolutionary Cost-Cognizant Regression Test Case Prioritization (ECRTP) and being implemented as regression testing approach for experiment.

***Index Terms*: Regression testing, Test case Prioritization, OOP, Cost-cognizant, APFDC.**

I. INTRODUCTION

Test case prioritization (TCP) is a software testing technique that finds an ideal ordering of test cases for regression testing, so that testers can obtain the maximum benefit of their test suite, even if the testing process is stop at some arbitrary point. The technique was first studied by [1]. Later, Sinha et al. [2] developed the technique in a more general context which was evaluated by [3]. Several TCP approaches are proposed in software testing community for regression testing. These approaches are evaluated and found to be efficiently and effectively revealing bugs earlier during regression testing. However, most of these approaches focus on procedural languages with only few on object-oriented programs [4]. Authors such as [5] and [6] addressed object-oriented programs but have the assumption that test case costs and fault severities are uniform. While in real sense, test case costs and fault severities vary [7]. Although some of these approaches [8], [9] used varying costs of test cases and severities of faults, but focused only on procedural programs. Moreover, most of these approaches adopted local search strategies to search for an optimized order of test cases for regression testing, meanwhile, these strategies mostly terminate at local optima [10] and [11]. Consequently, an evolutionary optimization technique based on genetic algorithm (GA) Mitchell [12] and Whitley [13] has been reported to produce an astonishingly better result when applied for propitiating test cases. In this paper, we proposed an evolutionary cost-cognizant regression TCP approach for OOP based on the use of the previous test case execution record and a GA. Tests costs, faults severities, and faults detected by each test case from the latest regression testing are gathered and then use a GA to find an order with the greatest rate of units of fault severity detected per unit test cost.

Revised Manuscript Received on September 22, 2019.

Abdulkarim Bello, Department of Computer Science and Information System, University Putra Malaysia, Serdang, Malaysia. kiyawa99@yahoo.com

Abu Bakar Md. Sultan, Department of Computer Science and Information System, University Putra Malaysia, Serdang, Malaysia. abakar@upm.edu.my

Abdul Azim Abdul Ghani, Department of Computer Science and Information System, University Putra Malaysia, Serdang, Malaysia. azim@upm.edu.my

Hazura Zulzalil, Department of Computer Science and Information System, University Putra Malaysia, Serdang, Malaysia. hazura@upm.edu.my

II. LITERATURE REVIEW

Test case prioritization approaches scheduled test cases in an order that increases their effectiveness at meeting some performance goal. To understand the progress of research in the field of regression testing prioritization, this section briefly presents prioritization techniques.

Goel & Sharma [14] proposed test case prioritization technique based on extended fault exposing potential. Their approach uses total and additional statement coverage and total and additional branch coverage exercised by each test case to measure the efficacy of the technique. Di Nardo et al. [15] proposed a test case prioritization technique based on four different coverage-based measures: Total Coverage; Additional Coverage; Total Coverage of modified code; and Additional Coverage of Modified Code. Hao et al [16] find out that most of the existing test case prioritization techniques separate the process of test case prioritization and the process of test case execution through the execution of all test cases before running the test cases. Therefore, they developed an adaptive test case prioritization technique that determines the execution order of test cases simultaneously during the execution of test cases.

Shahid & Ibrahim [17] proposed code coverage base regression test case prioritization technique that prioritizes test cases based on the percent of method covered by test cases. Zhang et al. [18] proposed test case prioritization technique that uses static paths on function call obtained by analyzing the source code, combined with the dynamic path after executing test cases, the correspondence is built between test cases and the static paths, identifying the changes which software developers modify program to correct defects, giving different priority to test case based on path coverage, test cases are selected in accordance with their priorities in regression testing.

Patil et al. [19] proposed the use of residual test coverage algorithm and statistical technique to develop test suite prioritization using residual coverage algorithm and statistical technique for both white-box and black-box testing. Miranda & Bertolino [20] proposed a new coverage-based test case prioritization technique using scope-aided testing approach. The technique considers reuse context while reordering test cases. Empirical evaluation of the technique reveals that the technique can improve the regression testing technique.

Panigrahi & Mall [21] proposed a heuristic based regression test case prioritization technique based on the analysis of dependence model for object-oriented programs. Huang et al. [22] used historical record of the latest regression testing on a GA to develop test case prioritization for regression testing to determine the most effective arrangement of test cases. Results of from evaluation of the technique of the technique indicated that the technique can improve the regression testing technique.

Li et al. [23] proposed an RTP approach that introduces the parallel fitness evaluation and parallel crossover. The approach uses computational scheme based on ordinal and sequential representation using a multi-objective evolutionary algorithm (NSG-II) to prioritize test suite. Empirical evaluation on eight benchmarks and one open source program show that a maximum speed-up is achieved. Panigrahi & Mall [24] proposed a test case prioritization technique for object-oriented programs. Velmurugan & Mahapatra [25] proposed a test case prioritization technique that considers branch coverage and DU pair coverage to improve the effectiveness of the regression testing process. The technique makes use of a Genetic Algorithm to prioritize test cases based on the fitness value of the individual test. Muthusamy & Kalimitu [26] proposed an evolutionary regression testing technique that prioritizes test cases base on the practical weights of ten factors that are grouped into four categories.

Our approach for automated regression test prioritization of OOPs was inspired by some of the above-mentioned works. For example, Elbaum et al. [27] and Malishevsky [7] defined a novel cost-cognizant test case prioritization approach for regression testing. This approach uses a systematic and well-coined process of assigning costs to test cases and severities to faults. However, the approach was later adopted by different researchers such as [8], [22], [28], [29] for developing different approaches to prioritize test cases during regression testing.

III. Evolutionary Cost-cognizant Test Case Selection and Prioritization (ECTSP)

The proposed ECRTP approach for object-oriented programs, as shown in the Fig. 1. The algorithm takes as input the source code of given object-oriented and test suite of the source program all written in Java programming language and return as the output prioritized test cases. Although ECRTP was tested on some specifically certain example of Java programs, it overall concepts can be applied to any other object-oriented language allowing developers to test their modified programs.

```
Algorithm: ECRTP
     input: Java Source code P, Test suite T of P
     output: prioritizedT
declaration:
begin  ECRTP
  1. Construct JSDG model M of P
  2. Create different faulty versions of P
  3. Update the JSDG model M
  4. Get all the affected statement
  5. Get the test case coverage information through path-based testing covInfo
  6. Identify and select the affected test cases
  7. Encode the selected test cases
  8. Generate initial population randomly
  9. Evaluate the fitness of the initial population
  10. Select two-best chromosomes for crossover
  11. WHILE NOT TERMINATION
  12.       Perform crossover on the two selected chromosomes
  13.       Perform mutation the chromosomes formed after crossover
  14.       Evaluate the fitness of the newly formed chromosomes
  15.       Add two-best chromosomes to the population
  16. return prioritizedT
end ECRTP
```

Fig. 1: ECRTP Algorithm Experimental Data

To analyze the effectiveness of fault detection for the Evolutionary Cost-cognizant Test Case Selection and Prioritization approach for Object-Oriented Programs with regards to the mutation score using different tests costs and fault severities (VcUs, UcVs, VcVs), experimental results of Table I was evaluated. The data contents of the table include the object programs (Object Programs), number of test cases (#Test Cases) that were selected from test suite of each object program, number of test cases that were executed (#Test Cas29
es) from among the selected test cases. The table also shows the number of killed mutants (#Killed Mutants), number of equivalent/undetected mutants (#Equivalent/Undetected Mutants), and the total number of mutants (#Mutants) for all the objects programs. The last column of the table shows the

effectiveness (EFFms(%)) of the mutation score for the different cost cognizant.

Experimental data presented in Table I was used to compute the fitness value by the employed GA using different test case costs uniform faults severity (VcUs), uniform test case cost different faults severities (UcVs), and different tests cost different faults severities (VcVs) respectively.

Table I: Experimental Data for the Cost-cognizant Approaches

Object Programs	# Test Cases	#Executed Test Cases	#Killed Mutants			#Equivalent Mutants			#Mutants	EFFms(%)		
			VcUs	UcVs	VcVs	VcUs	UcVs	VcVs		VcUs	UcVs	VcVs
Tri	6	2	58	58	61	42	42	39	100	58.00	58.00	61.00
BST1	20	4	45	48	64	43	40	24	88	51.14	54.55	72.73
BST2	34	3	68	70	70	28	26	26	96	70.83	72.92	72.92
BAC	16	3	45	50	45	55	50	55	100	45.00	50.00	45.00
SLL	18	2	53	50	53	27	30	27	80	66.25	62.50	66.25
ATRS	21	3	50	58	55	20	12	15	70	71.43	82.86	78.57
CFM	28	3	55	55	61	45	45	39	100	55.00	55.00	61.00
ECS	40	4	53	51	58	41	43	36	94	56.38	54.26	61.70
Total	185	24	427	440	467	301	288	261	728	474.00	490.07	519.17

The effectiveness of the three cost-cognizant test case prioritization approaches in terms of mutants' detectability was compared and discussed in this sub section. Charts from two figures were used to analyze the three approaches. **Error! Reference source not found.**. 2 compared the effectiveness of fault detection of the three approaches against the object programs used while **Error! Reference source not found.**3 compared the percentage of effectiveness of the approaches by considering the mutation score of the three cost-cognizant approaches.

IV. RESULTS ANALYSIS

From the result shown in **Error! Reference source not found.**, the overall fitness values of the effectiveness for the object programs were depicted in **Error! Reference source not found.**2. Four of the eight object programs that had the highest effectiveness values came from the three cost-cognizant approaches (VcUs, UcVs, VcVs). VcVs gets the highest effectiveness score on three object programs, it had highest effectiveness score on one object program (BST1, 72.73%) and sharing two highest scores with the remaining two approaches one program object to each of the remaining two approaches.

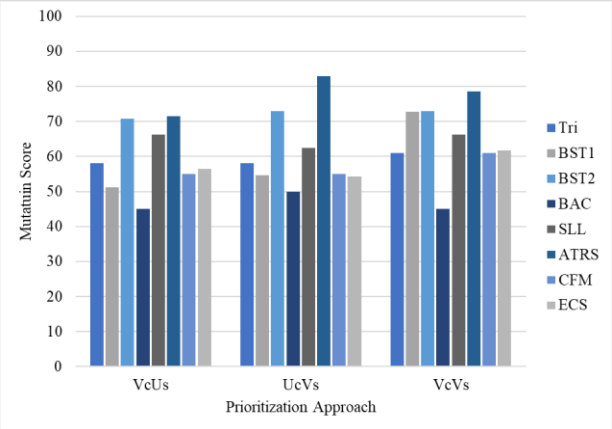

Fig. 2: Effectiveness of the Prioritization Approaches

UcVs gets the highest effectiveness score has the highest effectiveness score on one object program (ATRS, 82.86%) and sharing one highest score (BST2, 72.92%) with VcVs approach.

While the remaining highest effectiveness score among the scores of the four object programs (SLL, 66.25%) achieved by VcUs approach was also attained by VcVs approach. This made VcVs the most prepared cost-cognizant approach among the three approaches.

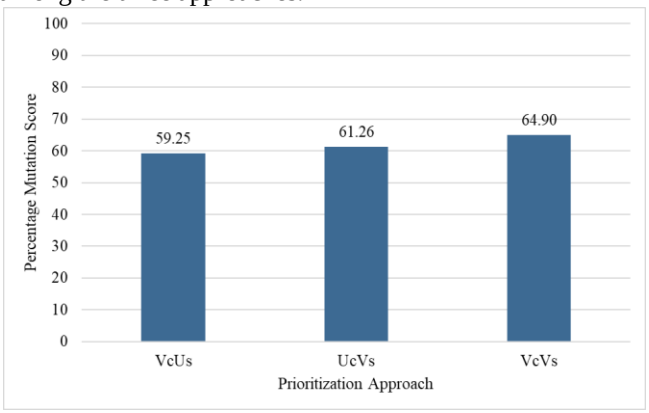

Fig. 3: Average Effectiveness of the Prioritization Approaches

Fig. 3 presents the average effectiveness score of fault detection for the three cost-cognizant prioritization approaches (VcUs, UcVs, VcVs). The chats depicted by this figure indicated that VcVs had the highest percentage score for the effectiveness (64.90%) of fault detection among the three cost-cognizant approaches.

UcVs gets the second highest effectiveness (61.26%) of fault detection, while the least score was attained by VcUs (59.25%).

V. DISCUSSION

From the analysis conducted on the experimental data presented in Table I and chats depicted by Fig. 2 and Fig. 3, it was observed that VcVs have the highest effectiveness of fault detection among the three cost-cognizant approaches. VcVs Scored the highest number of object programs' effectiveness of fault detection for the mutation score. The three of the four highest effectiveness attained by VcVs are BST1 72.73%, BST2 72.92%, and SLL 66.25%. While in terms of the percentage, the effectiveness of fault detection for the mutation score, VcVs had the highest effectiveness score among the three cost-cognizant prioritization approaches. Scoring 64.90% effectiveness of fault detection, which was the highest score attained by any of the three cost-cognizant prioritization approaches.

VI. CONCLUSION

Results of the experiment conducted, and the analysis presented on APFDc showed that there were statistical differences between ECRTP and other prioritization approaches or testing object-oriented programs. ECRTP was found to produces the best APFDc result as compared with the four other approaches followed by JaNaMa and then RanPrio on all the experimental objects. The last two approaches, NonPrio and RevPrio had the worst performance in terms of APFDc measure as they not able to produce a single best result on the eight experimental objects during empirical study. With the results of the experiment conducted, and the analysis presented on APFDc showed that there were statistical differences between ECRTP and other prioritization approaches or testing object-oriented programs. ECRTP was found to produces the best APFDc result as compared with the four other approaches followed by JaNaMa and then RanPrio on all the experimental objects. The last two approaches, NonPrio and RevPrio had the worst performance in terms of APFDc measure as they not able to produce a single best result on the eight experimental objects during empirical study.

ACKNOWLEDGMENT

We acknowledge that this research received support from the Fundamental Research Grant Scheme FRGS/1/2015/ICT01/UPM/02/12 awarded by Malaysian Ministry of Higher Education to the Faculty of Computer Science and Information Technology at University Putra Malaysia.

REFERENCES

1. W. E. Wong, J. R. Horgan, S. London, and A. P. Mathur, "Effect of Test Set Minimization on Fault Detection Effectiveness," *Softw. Pract. Exp.*, vol. 28, no. July 1996, pp. 347–369, Jul. 1998.
2. S. Sinha, M. J. Harrold, and G. Rothermel, "System-dependence-graph-based slicing of programs with arbitrary interprocedural control flow," *Softw. Eng. 1999. Proc. 1999 Int. Conf.*, vol. 1999, no. May 1999, pp. 432–441, May 1999.
3. G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, "Test Case Prioritization: an Empirical Study," *Proc. IEEE Int. Conf. Softw. Maint.*, p. 179, 1999.
4. J. Chen *et al.*, "Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering," *J. Syst. Softw.*, vol. 135, pp. 107–125, 2018.
5. S. Panda, D. Munjal, and D. P. Mohapatra, "A Slice-Based Change Impact Analysis for Regression Test Case Prioritization of Object-Oriented Programs," vol. 2016, pp. 1–25, 2016.
6. A. B. M. Sultan, A. A. Bin Abdul Ghani, S. Baharom, and S. Musa, "An Evolutionary Regression Test Case Prioritization based on Dependence Graph and Genetic Algorithm for," *Icetet*, pp. 22–26, 2014.
7. A. G. Malishevsky, J. R. Ruthruff, G. Rothermel, and S. Elbaum, "Cost-cognizant Test Case Prioritization," Department of Computer Science and Engineering University of NebraskaLincoln Techical Report, 2006.
8. M. Tulasiraman and V. Kalimuthu, "Cost Cognizant History Based Prioritization of Test Case for Regression Testing Using Immune Algorithm," *J. Intell. Eng. Syst.*, vol. 11, no. 1, pp. 221–228, 2018.
9. Y. Wang, X. Zhao, and X. Ding, "An Effective Test Case Prioritization Method Based on Fault Severity," in *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, 2015, vol. 2015–Novem, pp. 737–741.
10. A. B. Sanchez, S. Segura, and A. Ruiz-Cortes, "A Comparison of Test Case Prioritization Criteria for Software Product Lines," in *2014 IEEE Seventh International Conference on Software Testing, Verification and Validation*, 2014, pp. 41–50.
11. P. R. Srivastava and T. Kim, "Application of Genetic Algorithm in Software Testing," *Intenational J. Softw. Eng. Its Appl.*, vol. 3, no. 4, pp. 87–96, 2009.
12. M. Mitchell, *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*, Fifth Edit. The MIT Press (Fifth Edit). Cambridge, Massachusetts • London, England: Bradford Book The MIT Press, 1998.
13. D. Whitley, "An overview of evolutionary algorithms: Practical issues and common pitfalls," *Inf. Softw. Technol.*, vol. 43, no. 14, pp. 817–831, 2001.
14. N. Goel and M. Sharma, "Prioritization of Test Cases and Its Techniques," *Int. J. Comput. Appl.*, vol. 5, no. 4, pp. 85–89, 2015.
15. D. Di Nardo, N. Alshahwan, L. Briand, and Y. Labiche, "Coverage-based regression test case selection, minimization and prioritization: a case study on an industrial system," *Softw. Testing, Verif. Reliab.*, pp. 371–396, 2015.
16. D. Hao, X. Zhao, and L. Zhang, "Adaptive test-case prioritization guided by output inspection," in *Proceedings - International Computer Software and Applications Conference*, 2013, pp. 169–179.
17. M. Shahid and S. Ibrahim, "A New Code Based Test Case Prioritization Technique," vol. 8, no. 6, pp. 31–38, 2014.
18. Z. H. Zhang, Y. M. Mu, and Y. A. Tian, "Test case prioritization for regression testing based on function call path," in *Proceedings - 4th International Conference on Computational and Information Sciences, ICCIS 2012*, 2012, pp. 1372–1375.
19. H. Patil, G. A., A. H. Patil, N. Goveas, and K. Rangarajan, "N., & Rangarajan, K. (2016). Regression Test Suite Prioritization using Residual Test Coverage Algorithm and Statistical Techniques," *Int. J. Educ. Manag. Eng.*, vol. 6, no. 5, pp. 32–39, 2016.
20. B. Miranda and A. Bertolino, "Scope-aided Test Prioritization, Selection and Minimization for Software Reuse," *J. Syst. Softw.*, vol. 0, pp. 1–22, 2016.
21. C. R. Panigrahi and R. Mall, "A heuristic-based regression test case prioritization approach for object-oriented programs," *Innov. Syst. Softw. Eng.*, vol. 10, no. 3, pp. 155–163, 2014.
22. Y. C. Huang, K. L. Peng, and C. Y. Huang, "A history-based cost-cognizant test case prioritization technique in regression testing," *J. Syst. Softw.*, vol. 85, no. 3, pp. 626–637, 2012.
23. Z. Li, Y. Bian, R. Zhao, and J. Cheng, "A fine-grained parallel multi-objective test case prioritization on GPU," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8084 LNCS, pp. 111–125, 2013.
24. C. Rani Panigrahi and B. Rajib Mall, "An approach to prioritize the regression test cases of object-oriented programs," vol. 1, no. June, pp. 159–173, 2013.
25. P. Velmurugan and R. P. Mahapatra, "Effective Branch and DU Coverage Testing through Test Case Prioritization using Genetic Algorithm," *J. Theor. Appl. Inf. Technol.*, vol. 90, no. 1, 2016.
26. T. Muthusamy and S. K., "A Test Case Prioritization Method with Weight Factors in Regression Testing Based on Measurement Metrics," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 03, no. 12, pp. 390–396, 2013.
27. S. Elbaum, A. Malishevsky, and G. Rothermel, "Incorporating varying test costs and fault severities into test case prioritization," in *Proceedings - International Conference on Software Engineering*, 2001, pp. 329–338.
28. Y. C. Huang, C. Y. Huang, J. R. Chang, and T. Y. Chen, "Design and analysis of cost-cognizant test case prioritization using genetic algorithm with test history," *Proc. - Int. Comput. Softw. Appl. Conf.*, pp. 413–418, 2010.
29. H. Park, H. Ryu, and J. Baik, "Historical value-based approach for cost-cognizant test case prioritization to improve the effectiveness of regression testing," *Proc. - 2nd IEEE Int. Conf. Secur. Syst. Integr. Reliab. Improv.*, pp. 39–46, 2008.

AUTHORS PROFILE

Abdulkarim Bello is a Nigeria born, PhD student at the Department of Software Engineering and Information Systems Faculty of Computer Science and Information Technology University Putra Malaysia. Abdulkarim studied his BSc Computer Science at Usmanu Danfodiyo University Sokoto, Nigeria. He later proceeds to study his MSc Computer Science at Hamdard University New Delhi, India.

Prof. Dr. Abu Bakar Md. Sultan works with the University Putra Malaysia. Professor Dr. Abu Bakar Md Sultan studied his BSc in Comp. Sc. (ITM-UPM), M.Sc. (UPM), Ph.D. (UPM). As a Professor and former Dean faculty of Computer Science and Information Technology, University Putra Malaysia, have trained researchers in different software engineering disciplines that includes software testing, SQL injection vulnerability detection, regression testing etc.

Prof. Dr. Abdul Azim Abdul Ghani is a staff of the faculty of Computer Science and Information Technology University Putra Malaysia. Professor Dr. Abdul Azim Abdul Ghani studied his B.Sc. in Indiana State University, M.Sc. (Miami), PhD (Strathclyde). is Being an expert in Software Metrics, Agile Software Development and Aspect Oriented Software Testing, Prof. Dr. Abdul Azim Abdul Ghani have trained a lot of students and researchers in those areas.

Dr. Hazura Zulzalil is an Associate Professor at the Department of Software Engineering and Information Systems. Dr. Hazura Zulzalil studied her B.Sc., MSc and Ph. D. Computer Science in University Putra Malaysia (UPM). Dr Hazura Zulzali is an expert in Software Metrics, Software Quality and Software Engineering. Assoc. Prof. Dr. Hazura Zulzalil is the current Head, Department of Software Engineering and Information Systems, University Putra Malaysia.